Penkrot et al. 2018 Geosphere-Supplemental File 1

Table of Contents

01-Penkrot et al. 2018 Geosphere U1419 Supervised Classification Model

John M. Jaeger

March 20, 2018

The primary goal of this code is to test whether these sedimentary properties can discriminate amongst the different litofacies. This code will import elemental concentrations and physical property data from from Integrated Ocean Drilling Site U1419. See here for more details about this drilling location: http://iodp.tamu.edu/scienceops/expeditions/alaska_tectonics_climate.html

Code is available on Github at: https://github.com/jmjak86/Penkrot_et_al_2018_Geosphere

Code generated by: John M. Jaeger Associate Professor 241 Williamson Hall P.O. Box 112120 Dept. of Geological Sciences University of Florida Gainesville FL 32611-2120, USA (352) 846-1381 ORCID ID# orcid.org/0000-0003-0248-489X http://people.clas.ufl.edu/jmjaeger/

last edited: 20180320

Code Information

This code will import physical property data and normalized scanning XRF elemental concentrations (NMS normalized; Lyle et al., 2012) from Integrated Ocean Drilling Site U1419, and then perform two types of supevised classification on these data. See here for more details about this drilling location: http://iodp.tamu.edu/scienceops/expeditions/alaska_tectonics_climate.html.

- The datasets come from samples analyzed in the Department of Geological Sciences at the University of Florida and published values from: Walczak, M. H., Mix, A. C., Willse, T., Slagle, A., Stoner, J. S., Jaeger, J., … Kioka, A. (2015). Correction of non-intrusive drill core physical properties data for variability in recovered sediment volume. Geophysical Journal International, 202(2), 1317–1323. https://doi.org/10.1093/gji/ggv204

Load packages

```
library(plyr)
library(dplyr)
library(psych)
library(caret)
library(car)
library(robCompositions)
library(klaR)
library(e1071)
```

Import the data

```
# load the dataset
U1419_all <- read.csv("../raw_data/2018-03-20_U1419-Penkrot_Geosphere-2018-da
ta.csv")
U1419_all<-U1419_all[,1:15]
U1419.all<- tbl_df(U1419_all)
```

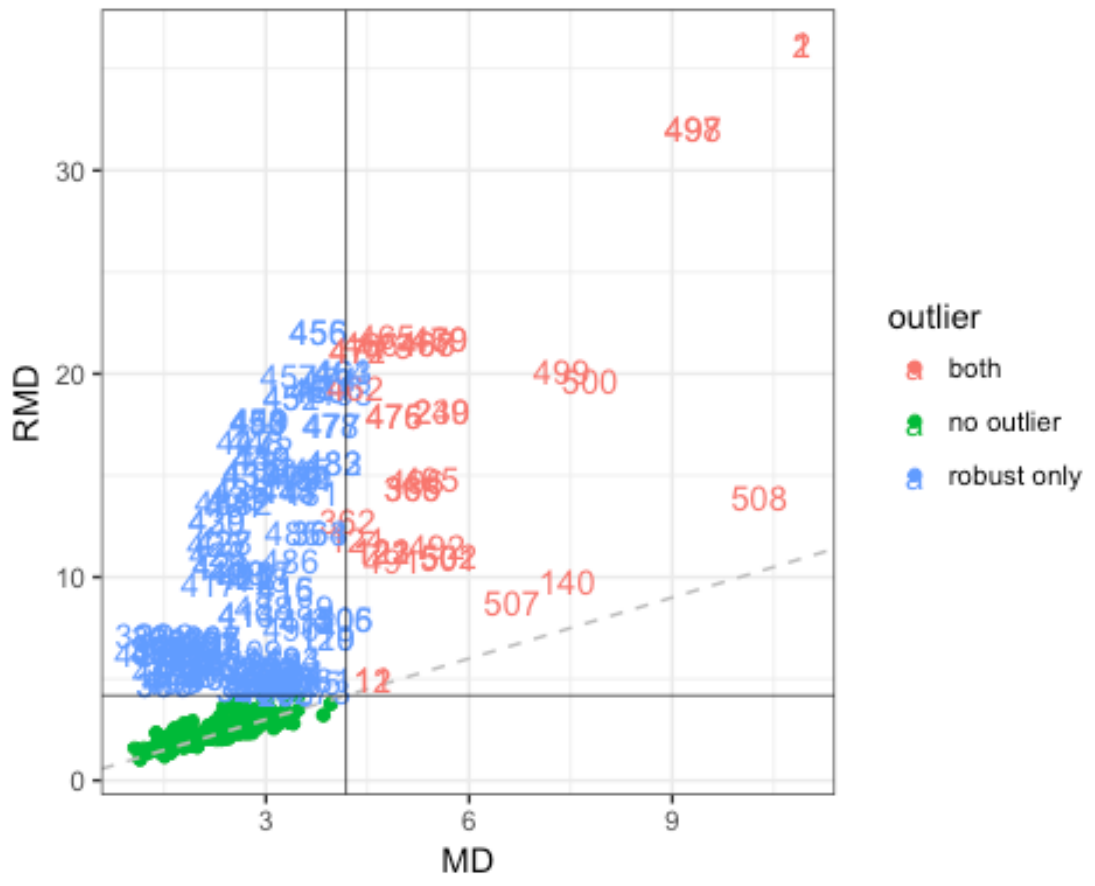
Data are in the following units (nms=normalized median-scaled method; Lyle et al., 2012):

Al (mass% nms);   Ca (mass% nms);   Zr (ppm nms); K (mass% nms);   Rb (mass% nms); Si (mass% nms);   b_star (unitless);   NGR (cps/g vol. normalized); MS (cm^3/g vol. normalized)

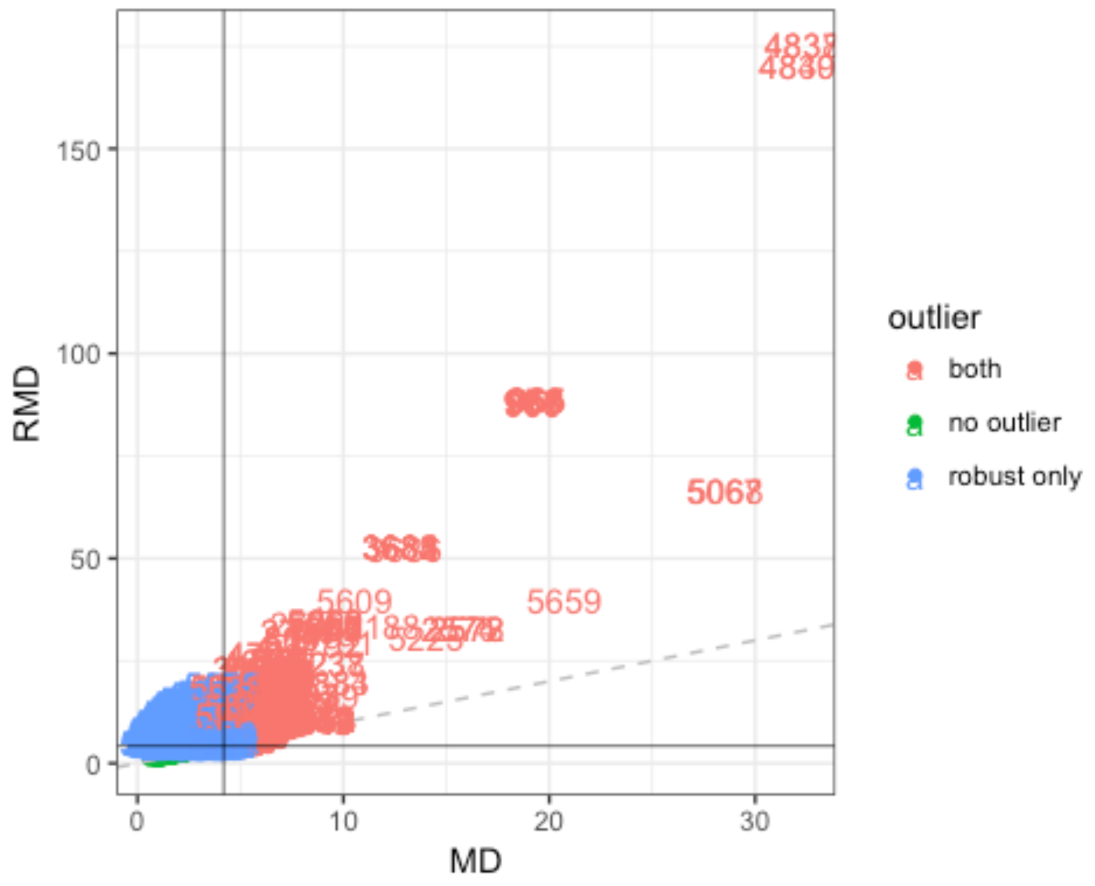
Data preparation

We remove outlier values because they have strong influence on variance-related analyses. Because the data include compositional parameters (i.e., elemental abundances), we use a robust routine from the robCompositions package (Filzmoser, P., & Hron, K. (2008). Outlier detection for compositional data using robust methods. Mathematical Geosciences, 40(3), 233–248. https://doi.org/10.1007/s11004-007-9141-5).

```
U1419.all[,11] <- U1419.all[,11]+10 # adjust b* so it has only positive value
s
nRows <- nrow(U1419.all)
nCols <- ncol(U1419.all)
U1419.mud <- U1419.all[1:508,]
U1419.diamict <- U1419.all[509:nRows,]

mudout <- outCoDa(U1419.mud[5:13], quantile = 0.975, method = "robust", h = 1
/2, coda=log)
plot(mudout,which=2)
```



```
mud.outclean <- U1419.mud[!mudout$outlierIndex,]
diaout <- outCoDa(U1419.diamict[5:13], quantile = 0.975, method = "robust", h
= 1/2, coda=log)
plot(diaout,which=2)
```



```
diamict.outclean <- U1419.diamict[!diaout$outlierIndex,]

U1419.allClean <- rbind(mud.outclean,diamict.outclean)
U1419.diamict <- U1419.allClean[509:nRows,]
```

Data Inspection

Near Zero Values

The next step is to remove any constant and almost constant predictors across samples (called zero and near-zero variance predictors) using the the function nearZeroVar from the caret package does. It not only removes predictors that have one unique value across samples (zero variance predictors), but also removes predictors that have both 1) few unique values relative to the number of samples and 2) large ratio of the frequency of the most common value to the frequency of the second most common value (near-zero variance predictors).

```
nearZeroVar(U1419.allClean[,5:13], saveMetrics = TRUE)

##        freqRatio percentUnique zeroVar   nzv
## Al      1.454545    43.8385573   FALSE FALSE
## Ca      1.400000    57.1275225   FALSE FALSE
## Zr      1.200000    59.1670245   FALSE FALSE
## K       1.500000    40.1674538   FALSE FALSE
## Rb      1.200000    57.3207385   FALSE FALSE
## Si      1.090909    49.0768570   FALSE FALSE
## b_star  1.083871     1.6745384   FALSE FALSE
## NGR     1.656194     0.2146844   FALSE FALSE
## MS      1.055556    11.0991842   FALSE FALSE
```

```
nearZeroVar(U1419.diamict[,5:13], saveMetrics = TRUE)
```

```
##         freqRatio percentUnique zeroVar   nzv
## Al      1.454545    29.6890672   FALSE FALSE
## Ca      1.400000    39.9699097   FALSE FALSE
## Zr      1.200000    41.6248746   FALSE FALSE
## K       1.500000    28.7362086   FALSE FALSE
## Rb      1.200000    39.5854229   FALSE FALSE
## Si      1.090909    33.7345369   FALSE FALSE
## b_star  1.058442     1.1200267   FALSE FALSE
## NGR     1.720299     0.1170177   FALSE FALSE
## MS      1.055556     7.5727182   FALSE FALSE
```

The results show that all variables do not have zer-zero variance so they are all included in the following data preparation steps.

Kruskal-Wallis tests

We first perform a non-parametric Kruskal-Wallis test on each property to see if the Mud and Diamict lithofacies have unique values following Collins, A. L., Walling, D. E., & Leeks, G. J. L. (1998). Use of composite fingerprints to determine the provenance of the contemporary suspended sediment load transported by rivers. Earth Surface Processes and Landforms, 23(1), 31–52. https://doi.org/10.1002/(SICI)1096-9837(199801)23:1<31::AID-ESP816>3.0.CO;2-Z

```
#Kruskal-Wallis test to see if lithofacies are unique; if P-value <<.05 signi
ficance level, we conclude that samples are nonidentical populations.

U1419.allClean$Mud_Diamict <- as.factor(U1419.allClean$Mud_Diamict)

kruskal.test(U1419.allClean$Al, U1419.allClean$Mud_Diamict)

##
##  Kruskal-Wallis rank sum test
##
## data:  U1419.allClean$Al and U1419.allClean$Mud_Diamict
## Kruskal-Wallis chi-squared = 817.88, df = 1, p-value < 2.2e-16

kruskal.test(U1419.allClean$Ca, U1419.allClean$Mud_Diamict)

##
##  Kruskal-Wallis rank sum test
##
```

```
## data:  U1419.allClean$Ca and U1419.allClean$Mud_Diamict
## Kruskal-Wallis chi-squared = 300.02, df = 1, p-value < 2.2e-16

kruskal.test(U1419.allClean$Zr, U1419.allClean$Mud_Diamict)

##
##  Kruskal-Wallis rank sum test
##
## data:  U1419.allClean$Zr and U1419.allClean$Mud_Diamict
## Kruskal-Wallis chi-squared = 6.9157, df = 1, p-value = 0.008544

kruskal.test(U1419.allClean$K, U1419.allClean$Mud_Diamict)

##
##  Kruskal-Wallis rank sum test
##
## data:  U1419.allClean$K and U1419.allClean$Mud_Diamict
## Kruskal-Wallis chi-squared = 210.07, df = 1, p-value < 2.2e-16

kruskal.test(U1419.allClean$Rb, U1419.allClean$Mud_Diamict)

##
##  Kruskal-Wallis rank sum test
##
## data:  U1419.allClean$Rb and U1419.allClean$Mud_Diamict
## Kruskal-Wallis chi-squared = 847.85, df = 1, p-value < 2.2e-16

kruskal.test(U1419.allClean$Si, U1419.allClean$Mud_Diamict)

##
##  Kruskal-Wallis rank sum test
##
## data:  U1419.allClean$Si and U1419.allClean$Mud_Diamict
## Kruskal-Wallis chi-squared = 764.27, df = 1, p-value < 2.2e-16

kruskal.test(U1419.allClean$b_star, U1419.allClean$Mud_Diamict)

##
##  Kruskal-Wallis rank sum test
##
## data:  U1419.allClean$b_star and U1419.allClean$Mud_Diamict
## Kruskal-Wallis chi-squared = 804.61, df = 1, p-value < 2.2e-16

kruskal.test(U1419.allClean$NGR, U1419.allClean$Mud_Diamict)

##
##  Kruskal-Wallis rank sum test
##
## data:  U1419.allClean$NGR and U1419.allClean$Mud_Diamict
## Kruskal-Wallis chi-squared = 953.45, df = 1, p-value < 2.2e-16

kruskal.test(U1419.allClean$MS, U1419.allClean$Mud_Diamict)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  U1419.allClean$MS and U1419.allClean$Mud_Diamict
## Kruskal-Wallis chi-squared = 570.42, df = 1, p-value < 2.2e-16
```

All sedimentary properties are distinctive at p=0.05 for the Mud-Diamict binary lithofacies model.

We now repeat the same test for the Diamict-only samples.

```
#Kruskal-Wallis test to see if lithofacies are unique; if P-value <<.05 signi
ficance level, we conclude that samples are nonidentical populations.

U1419.allClean$Diamict_only <- as.factor(U1419.allClean$Diamict_only)

kruskal.test(U1419.diamict$Al, U1419.diamict$Diamict_only)

##
##  Kruskal-Wallis rank sum test
##
## data:  U1419.diamict$Al and U1419.diamict$Diamict_only
## Kruskal-Wallis chi-squared = 289.59, df = 4, p-value < 2.2e-16

kruskal.test(U1419.diamict$Ca, U1419.diamict$Diamict_only)

##
##  Kruskal-Wallis rank sum test
##
## data:  U1419.diamict$Ca and U1419.diamict$Diamict_only
## Kruskal-Wallis chi-squared = 99.87, df = 4, p-value < 2.2e-16

kruskal.test(U1419.diamict$Zr, U1419.diamict$Diamict_only)

##
##  Kruskal-Wallis rank sum test
##
## data:  U1419.diamict$Zr and U1419.diamict$Diamict_only
## Kruskal-Wallis chi-squared = 367.27, df = 4, p-value < 2.2e-16

kruskal.test(U1419.diamict$K, U1419.diamict$Diamict_only)

##
##  Kruskal-Wallis rank sum test
##
## data:  U1419.diamict$K and U1419.diamict$Diamict_only
## Kruskal-Wallis chi-squared = 368.06, df = 4, p-value < 2.2e-16

kruskal.test(U1419.diamict$Rb, U1419.diamict$Diamict_only)

##
##  Kruskal-Wallis rank sum test
```

```
##
## data:  U1419.diamict$Rb and U1419.diamict$Diamict_only
## Kruskal-Wallis chi-squared = 401.9, df = 4, p-value < 2.2e-16

kruskal.test(U1419.diamict$Si, U1419.diamict$Diamict_only)

##
##  Kruskal-Wallis rank sum test
##
## data:  U1419.diamict$Si and U1419.diamict$Diamict_only
## Kruskal-Wallis chi-squared = 614.04, df = 4, p-value < 2.2e-16

kruskal.test(U1419.diamict$b_star, U1419.diamict$Diamict_only)

##
##  Kruskal-Wallis rank sum test
##
## data:  U1419.diamict$b_star and U1419.diamict$Diamict_only
## Kruskal-Wallis chi-squared = 218.76, df = 4, p-value < 2.2e-16

kruskal.test(U1419.diamict$NGR, U1419.diamict$Diamict_only)

##
##  Kruskal-Wallis rank sum test
##
## data:  U1419.diamict$NGR and U1419.diamict$Diamict_only
## Kruskal-Wallis chi-squared = 226.42, df = 4, p-value < 2.2e-16

kruskal.test(U1419.diamict$MS, U1419.diamict$Diamict_only)

##
##  Kruskal-Wallis rank sum test
##
## data:  U1419.diamict$MS and U1419.diamict$Diamict_only
## Kruskal-Wallis chi-squared = 86.652, df = 4, p-value < 2.2e-16
```

All sedimentary properties are distinctive at p=0.05 for the Diamict-only lithofacies model.

Data Correlation

The next step is to eliminate highly correlated elements following methodology here: https://topepo.github.io/caret/pre-processing.html

```
#Identifying Correlated Predictors in Mud/diamict data
AllCor <-  cor(U1419.allClean[,5:13],use="pairwise.complete.obs")
hc <- findCorrelation(as.matrix(AllCor), cutoff=0.9) # putt any value as a "c
utoff"
hc <- sort(hc)
print(AllCor)

##                 Al          Ca         Zr           K         Rb
## Al      1.00000000 -0.69620646 -0.5278694 -0.10235944 -0.6759842
## Ca     -0.69620646  1.00000000  0.3409005 -0.08431194  0.3000243
## Zr     -0.52786938  0.34090055  1.0000000  0.12035672  0.2688239
## K      -0.10235944 -0.08431194  0.1203567  1.00000000  0.6701262
## Rb     -0.67598425  0.30002426  0.2688239  0.67012619  1.0000000
## Si      0.86703600 -0.44302572 -0.4448555 -0.30042874 -0.7618901
## b_star -0.62311395  0.47376708  0.3225605  0.25490363  0.5762382
## NGR     0.72681284 -0.52669585 -0.3173942 -0.09539348 -0.5861278
## MS      0.02359015  0.39456129  0.1292273 -0.28970024 -0.3570453
##               Si     b_star         NGR          MS
## Al      0.8670360 -0.6231140  0.72681284  0.02359015
## Ca     -0.4430257  0.4737671 -0.52669585  0.39456129
## Zr     -0.4448555  0.3225605 -0.31739421  0.12922734
## K      -0.3004287  0.2549036 -0.09539348 -0.28970024
## Rb     -0.7618901  0.5762382 -0.58612775 -0.35704530
## Si      1.0000000 -0.5021288  0.65402498  0.22079750
## b_star -0.5021288  1.0000000 -0.60724293 -0.13292707
## NGR     0.6540250 -0.6072429  1.00000000  0.21907961
## MS      0.2207975 -0.1329271  0.21907961  1.00000000

if(length(hc)==0){
  print(AllCor)
}else{
  print(AllCor[-hc,-hc])
}

##                 Al          Ca         Zr           K         Rb
## Al      1.00000000 -0.69620646 -0.5278694 -0.10235944 -0.6759842
## Ca     -0.69620646  1.00000000  0.3409005 -0.08431194  0.3000243
## Zr     -0.52786938  0.34090055  1.0000000  0.12035672  0.2688239
## K      -0.10235944 -0.08431194  0.1203567  1.00000000  0.6701262
## Rb     -0.67598425  0.30002426  0.2688239  0.67012619  1.0000000
## Si      0.86703600 -0.44302572 -0.4448555 -0.30042874 -0.7618901
## b_star -0.62311395  0.47376708  0.3225605  0.25490363  0.5762382
## NGR     0.72681284 -0.52669585 -0.3173942 -0.09539348 -0.5861278
## MS      0.02359015  0.39456129  0.1292273 -0.28970024 -0.3570453
##               Si     b_star         NGR          MS
## Al      0.8670360 -0.6231140  0.72681284  0.02359015
## Ca     -0.4430257  0.4737671 -0.52669585  0.39456129
## Zr     -0.4448555  0.3225605 -0.31739421  0.12922734
## K      -0.3004287  0.2549036 -0.09539348 -0.28970024
## Rb     -0.7618901  0.5762382 -0.58612775 -0.35704530
## Si      1.0000000 -0.5021288  0.65402498  0.22079750
## b_star -0.5021288  1.0000000 -0.60724293 -0.13292707
## NGR     0.6540250 -0.6072429  1.00000000  0.21907961
## MS      0.2207975 -0.1329271  0.21907961  1.00000000

# no parameters are correlated at >0.9

#Identifying Correlated Predictors in Diamict-only data
DiaCor <-  cor(U1419.diamict[,5:13],use="pairwise.complete.obs")
```

```
hc2 <- findCorrelation(as.matrix(DiaCor), cutoff=0.9) # putt any value as a "
cutoff"
hc2 <- sort(hc2)
print(DiaCor)
```

```
##                 Al         Ca         Zr           K            Rb
## Al      1.00000000 -0.6913392 -0.5779896  0.06702474 -3.951456e-01
## Ca     -0.69133915  1.0000000  0.2943595 -0.11846426  1.802858e-01
## Zr     -0.57798957  0.2943595  1.0000000  0.14693810  3.267864e-01
## K       0.06702474 -0.1184643  0.1469381  1.00000000  7.528147e-01
## Rb     -0.39514557  0.1802858  0.3267864  0.75281467  1.000000e+00
## Si      0.75845747 -0.3260279 -0.4785979 -0.19106541 -5.332767e-01
## b_star -0.38320532  0.3731763  0.3685637  0.20181033  3.063185e-01
## NGR     0.48543376 -0.4893586 -0.4065281  0.10395204 -1.081213e-01
## MS     -0.35684211  0.5875183  0.1153821 -0.14750124 -3.230785e-05
##                Si     b_star        NGR            MS
## Al      0.7584575 -0.3832053  0.4854338 -3.568421e-01
## Ca     -0.3260279  0.3731763 -0.4893586  5.875183e-01
## Zr     -0.4785979  0.3685637 -0.4065281  1.153821e-01
## K      -0.1910654  0.2018103  0.1039520 -1.475012e-01
## Rb     -0.5332767  0.3063185 -0.1081213 -3.230785e-05
## Si      1.0000000 -0.1941714  0.3616576 -7.623560e-02
## b_star -0.1941714  1.0000000 -0.2552473  1.586736e-01
## NGR     0.3616576 -0.2552473  1.0000000 -1.644191e-01
## MS     -0.0762356  0.1586736 -0.1644191  1.000000e+00
```

```
if(length(hc2)==0){
  print(DiaCor)
}else{
  print(DiaCor[-hc2,-hc2])
}
```

```
##                 Al         Ca         Zr           K            Rb
## Al      1.00000000 -0.6913392 -0.5779896  0.06702474 -3.951456e-01
## Ca     -0.69133915  1.0000000  0.2943595 -0.11846426  1.802858e-01
## Zr     -0.57798957  0.2943595  1.0000000  0.14693810  3.267864e-01
## K       0.06702474 -0.1184643  0.1469381  1.00000000  7.528147e-01
## Rb     -0.39514557  0.1802858  0.3267864  0.75281467  1.000000e+00
## Si      0.75845747 -0.3260279 -0.4785979 -0.19106541 -5.332767e-01
## b_star -0.38320532  0.3731763  0.3685637  0.20181033  3.063185e-01
## NGR     0.48543376 -0.4893586 -0.4065281  0.10395204 -1.081213e-01
## MS     -0.35684211  0.5875183  0.1153821 -0.14750124 -3.230785e-05
##                Si     b_star        NGR            MS
## Al      0.7584575 -0.3832053  0.4854338 -3.568421e-01
## Ca     -0.3260279  0.3731763 -0.4893586  5.875183e-01
## Zr     -0.4785979  0.3685637 -0.4065281  1.153821e-01
## K      -0.1910654  0.2018103  0.1039520 -1.475012e-01
## Rb     -0.5332767  0.3063185 -0.1081213 -3.230785e-05
## Si      1.0000000 -0.1941714  0.3616576 -7.623560e-02
## b_star -0.1941714  1.0000000 -0.2552473  1.586736e-01
## NGR     0.3616576 -0.2552473  1.0000000 -1.644191e-01
## MS     -0.0762356  0.1586736 -0.1644191  1.000000e+00

# no parameters are significantly correlated
```

No parameters are positvely correlated at >0.9 so all will be used in subsequent analyses.

Levine Test for data distribution

The next processing step is to determine which type of discriminant analyses to conduct (linear or quadratic) to test for discrimation power of the chosen properties This is done with the Levene test, which is less sensistive to non-normality of data following methods here: http://www.itl.nist.gov/div898/handbook/eda/section3/eda35a.htm. For this section, only one element is analyzed (Ca). Replace Ca with other parameters to test.

```
## Tests for Homogeneity of variance, run for each element ------------------
---------------------------------
leveneTest(Ca~ Mud_Diamict, data=U1419.allClean)#p value >0.05 means they are
homogeneous; if not homogeneous, you cannot use LDA

## Levene's Test for Homogeneity of Variance (center = median)
##          Df F value    Pr(>F)    
## group    1  11.982 0.0005419 ***
##        4656                      
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Ca, Zr, K, Rb, Si, MS do not have homogeneous variance in the mud-diamict dataset, so a quadratic discrimination analysis will be used

```
## Tests for Homogeneity of variance, run for each element ------------------
---------------------------------
leveneTest(Ca~ Diamict_only, data=U1419.diamict)#p value >0.05 means they are
homogeneous; if not homogeneous, you cannot use LDA

## Levene's Test for Homogeneity of Variance (center = median)
##          Df F value    Pr(>F)    
## group    4  8.9696 3.284e-07 ***
##        4145                      
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Al, Zr, K, Rb, Si, b_star, NGR, MS do not have homogeneous variance in the diamict-only dataset, so a quadratic discrimination analysis will be used

Data Transformation-Full Dataset

The next step is to transform and scale data. A center-log ratio transformation is chosen for elemental data to remove constant-sum effects (Templ, M., Filzmoser, P., & Reimann, C. (2008). Cluster analysis applied to regional geochemical data: Problems and possibilities.

Applied Geochemistry, 23(8), 2198–2213.
https://doi.org/10.1016/j.apgeochem.2008.03.004).

```
mudout <- U1419.allClean[which(U1419.allClean$Mud_Diamict== "mud"), ]
diamictout <- U1419.allClean[which(U1419.allClean$Mud_Diamict== "diamict"), ]

dataCLRmud <- cenLR(mudout[,5:10]) #clr transformed elemental data
dataCLRdiamict <- cenLR(diamictout[,5:10]) #clr transformed elemmental data

mudCLR <- dataCLRmud$x.clr
diamictCLR <- dataCLRdiamict$x.clr

allelementdata.CLR <- rbind(mudCLR,diamictCLR)
ppdata <- U1419.allClean[,13:nCols-2]

allelements.norm <- apply(allelementdata.CLR, MARGIN = 2, FUN = function(X) (
X - min(X))/diff(range(X)))
ppdataA.norm  <- apply(U1419.allClean[,13:nCols-2], MARGIN = 2, FUN = functio
n(X) (X - min(X))/diff(range(X)))

lith1 <- U1419.allClean[,3]
lith2 <- U1419.allClean[,4]
alldata.norm<- cbind(lith1,allelements.norm,ppdataA.norm)

diamictdata<- cbind(lith2,allelementdata.CLR,ppdata)
nRows <- nrow(diamictdata)
diamictdataD<- diamictdata[509:nRows,]
diamictdata.norm1 <- apply(diamictdataD[,2:10], MARGIN = 2, FUN = function(X)
(X - min(X))/diff(range(X)))
diamictdata.norm <- cbind(lith2[509:nRows,],diamictdata.norm1)
```

Quadradic Discrimination Analysis of Mud-Diamict Lithofacies

The next step is to select the elements that best discriminate amongst the groups. The Greedy Wilks approach is used following methods of Gorman Sanisaca, L.E., Gellis, A.C., and Lorenz, D.L., 2017, Determining the sources of fine-grained sediment using the Sediment Source Assessment Tool (Sed_SAT): U.S. Geological Survey Open File Report 2017–1062, 104 p., https://doi.org/10.3133/ofr20171062

```
gw_obj<- greedy.wilks(Mud_Diamict~., data=alldata.norm[,1:10], niveau = 0.05)
## 'niveau' is probabilty that addition of variable does not contribute to mo
del
gw_obj

## Formula containing included variables:
##
## Mud_Diamict ~ Al + NGR + Rb + Si + MS + b_star + Ca
## <environment: 0x7f93e3456eb0>
##
```

```
##
## Values calculated in each step of the selection procedure:
##
##     vars Wilks.lambda F.statistics.overall p.value.overall
## 1     Al    0.6720183             2272.383               0
## 2    NGR    0.4488721             2857.718               0
## 3     Rb    0.2863132             3866.975               0
## 4     Si    0.2433640             3616.628               0
## 5     MS    0.2289303             3133.719               0
## 6 b_star    0.2177738             2784.337               0
## 7     Ca    0.2116233             2474.716               0
##   F.statistics.diff p.value.diff
## 1         2272.3832            0
## 2         2314.1229            0
## 3         2642.3832            0
## 4          821.1684            0
## 5          293.3002            0
## 6          238.2705            0
## 7          135.1457            0
```

The next step is to test the discrimination power of the combination of the elements chosen from the Greedy Wilks routine.

```
Group_mod1All<-dplyr::select(alldata.norm,Mud_Diamict,Al,NGR,Rb,Si,MS,b_star,
Ca)
# Select Training and Testing subsets -------------------------------------
data <- Group_mod1All
nColsF <- ncol(data)
set.seed(2969)
sample.ind = sample(2,
                    nrow(data),
                    replace = T,
                    prob = c(0.25,0.75))
data.test = data[sample.ind==1,]#data.dev
data.train = data[sample.ind==2,]#data.val
dataD.train <-data.train
#See how balanced the test & training sets look as Group; training set should
be balanced
table(data$Mud_Diamict)/nrow(data)

##
##    diamict        mud
## 0.93237441 0.06762559

table(data.test$Mud_Diamict)/nrow(data.test)

##
##   diamict       mud
## 0.9374457 0.0625543

table(data.train$Mud_Diamict)/nrow(data.train)
```

```
##
##    diamict        mud
## 0.93071001 0.06928999
```

```
#if one of the training groups is too large, you need to resample using Caret
set.seed(9560)
dtrainCols <- ncol(data.train)

down_train <- downSample(x = data.train[, 1:dtrainCols],
                         y = data.train$Mud_Diamict)
table(down_train$Mud_Diamict)/nrow(down_train)
```

```
##
## diamict     mud
##     0.5     0.5
```

```
dataD.train <- down_train[,1:dtrainCols]

# Discriminant Testing ---------------------------------------------------
qda.fit <- qda(Mud_Diamict ~., data=dataD.train)
qda.fit
```

```
## Call:
## qda(Mud_Diamict ~ ., data = dataD.train)
##
## Prior probabilities of groups:
## diamict     mud
##     0.5     0.5
##
## Group means:
##                Al       NGR        Rb        Si        MS    b_star
## diamict 0.7477778 0.6451760 0.3071679 0.7141303 0.4223308 0.3603694
## mud     0.3994768 0.1518061 0.6914481 0.4236820 0.1547725 0.7535171
##                Ca
## diamict 0.4145803
## mud     0.5625050
```

```
qda.class <- predict(qda.fit, data.test)$class
qda1.table <- table(qda.class, data.test$Mud_Diamict)
qda1.table
```

```
##
## qda.class diamict  mud
##   diamict    1079    0
##   mud           0   72
```

```
mean(qda.class == data.test$Mud_Diamict)
```

```
## [1] 1
```

```
fitControl <- trainControl(## 10-fold CV
  method = "repeatedcv",
  number = 10,
  ## repeated ten times
  repeats = 100)
set.seed(825)
qdaFit1 <- train(Mud_Diamict~ ., data = dataD.train,
                 method = "qda",
                 trControl = fitControl,
                 finalModel=TRUE,
                 verbose = FALSE,
                 na.action = na.omit)
qdaFit1

## Quadratic Discriminant Analysis
##
## 486 samples
##   7 predictor
##   2 classes: 'diamict', 'mud'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 100 times)
## Summary of sample sizes: 437, 438, 437, 438, 437, 438, ...
## Resampling results:
##
##   Accuracy  Kappa
##   1         1

qdaFit1$finalModel

## Call:
## qda(x, grouping = y, finalModel = TRUE, verbose = FALSE)
##
## Prior probabilities of groups:
## diamict     mud
##     0.5     0.5
##
## Group means:
##                Al       NGR        Rb        Si        MS    b_star
## diamict 0.7477778 0.6451760 0.3071679 0.7141303 0.4223308 0.3603694
## mud     0.3994768 0.1518061 0.6914481 0.4236820 0.1547725 0.7535171
##                Ca
## diamict 0.4145803
## mud     0.5625050

confusionMatrix(data.test$Mud_Diamict, predict(qdaFit1, data.test))

## Confusion Matrix and Statistics
##
##           Reference
## Prediction diamict  mud
##    diamict    1079    0
##    mud           0   72
```

```
##
##                Accuracy : 1
##                  95% CI : (0.9968, 1)
##     No Information Rate : 0.9374
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.9374
##          Detection Rate : 0.9374
##    Detection Prevalence : 0.9374
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : diamict
##
```

```
#relative importance of variables in model
qdaImp <- varImp(qdaFit1, scale = FALSE)
qdaImp
```

```
## ROC curve variable importance
##
##        Importance
## NGR        0.9980
## Rb         0.9928
## b_star     0.9761
## Al         0.9684
## Si         0.9389
## MS         0.9056
## Ca         0.7721
```

Support Vector Machine Classification Analysis of Mud-Diamict Lithofacies

The SVM design follows Masaaki Tsujitani and Yusuke Tanaka, “Cross-Validation, Bootstrap, and Support Vector Machines,” Advances in Artificial Neural Systems, vol. 2011, Article ID 302572, 6 pages, 2011. doi:10.1155/2011/302572

```
Group_mod1All<-dplyr::select(alldata.norm,Mud_Diamict,Al,NGR,Rb,Si,MS,b_star,
Ca)

## svm >
svm.model1 <- svm(Mud_Diamict~ ., data = dataD.train, cost = 100, gamma = 1)
svm.pred1 <- predict(svm.model1, data.test)
## compute svm confusion matrix >
```

```
svmtable1 <- table(pred = svm.pred1, data.test$Mud_Diamict)
svmtable1

##
## pred      diamict  mud
##   diamict    1079    0
##   mud           0   72

classAgreement(svmtable1,qda1.table)

## $diag
## [1] 1
##
## $kappa
## [1] 1
##
## $rand
## [1] 1
##
## $crand
## [1] 1

## compute rpart confusion matrix >
fitControl <- trainControl(## 10-fold CV
  method = "repeatedcv",
  number = 10,#10
  ## repeated ten times
  repeats = 100)#100
set.seed(645)
#fitControl <- trainControl(number = 200)
SVMFit1 <- train(Mud_Diamict~ ., data = dataD.train,
                 method = "svmRadial",
                 trControl = fitControl,
                 tuneLength = 3,
                 finalModel=TRUE,
                 verbose = FALSE,
                 scaled = FALSE,
                 na.action = na.omit)

## Loading required package: kernlab

##
## Attaching package: 'kernlab'

## The following object is masked from 'package:ggplot2':
##
##     alpha

## The following object is masked from 'package:psych':
##
##     alpha
```

```
SVMFit1

## Support Vector Machines with Radial Basis Function Kernel
##
## 486 samples
##   7 predictor
##   2 classes: 'diamict', 'mud'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 100 times)
## Summary of sample sizes: 438, 438, 437, 438, 437, 438, ...
## Resampling results across tuning parameters:
##
##   C     Accuracy   Kappa
##   0.25  0.9999792  0.9999583
##   0.50  1.0000000  1.0000000
##   1.00  1.0000000  1.0000000
##
## Tuning parameter 'sigma' was held constant at a value of 0.3479712
## Accuracy was used to select the optimal model using  the largest value.
## The final values used for the model were sigma = 0.3479712 and C = 0.5.

SVMFit1$finalModel

## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 0.5
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.347971239344602
##
## Number of Support Vectors : 61
##
## Objective Function Value : -17.5037
## Training error : 0

confusionMatrix(data.test$Mud_Diamict, predict(SVMFit1, data.test))

## Confusion Matrix and Statistics
##
##             Reference
## Prediction diamict  mud
##    diamict    1079    0
##    mud           0   72
##
##                Accuracy : 1
##                  95% CI : (0.9968, 1)
##     No Information Rate : 0.9374
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                    Kappa : 1
##   Mcnemar's Test P-Value : NA
##
##              Sensitivity : 1.0000
##              Specificity : 1.0000
##           Pos Pred Value : 1.0000
##           Neg Pred Value : 1.0000
##               Prevalence : 0.9374
##           Detection Rate : 0.9374
##     Detection Prevalence : 0.9374
##        Balanced Accuracy : 1.0000
##
##         'Positive' Class : diamict
##

#relative importance of variables in model
SVMImp1 <- varImp(SVMFit1, scale = FALSE)
SVMImp1

## ROC curve variable importance
##
##         Importance
## NGR         0.9980
## Rb          0.9928
## b_star      0.9761
## Al          0.9684
## Si          0.9389
## MS          0.9056
## Ca          0.7721
```

Quadradic Discrimination Analysis of Diamict-only Lithofacies

```
diamictsel <- subset(diamictdata.norm, select=c("Diamict_only", "Al", "Ca","Z
r","Rb","Si","MS","b_star","NGR")) #K and Rb are too correlated and routine c
rashes; Rb only is used

gw_obj2<- greedy.wilks(Diamict_only~., data=diamictsel, niveau = 0.05)## 'niv
eau' is probabilty that addition of variable does not contribute to model
gw_obj2

## Formula containing included variables:
##
## Diamict_only ~ Si + Zr + b_star + Al + NGR + MS + Ca + Rb
## <environment: 0x7f93c6604b80>
##
##
## Values calculated in each step of the selection procedure:
##
##     vars Wilks.lambda F.statistics.overall p.value.overall
## 1     Si    0.8337180            206.67628   7.093025e-162
## 2     Zr    0.7345493            172.78561   8.659758e-271
## 3 b_star    0.6843991            140.77478    0.000000e+00
## 4     Al    0.6555591            117.23534    0.000000e+00
## 5    NGR    0.6367466            100.12423    0.000000e+00
## 6     MS    0.6203556             88.27207    0.000000e+00
## 7     Ca    0.6123183             77.67767    0.000000e+00
## 8     Rb    0.5983227             71.27323    0.000000e+00
##   F.statistics.diff  p.value.diff
## 1         206.67628 7.093025e-162
## 2         139.86639  0.000000e+00
## 3          75.89582  0.000000e+00
## 4          45.55482  0.000000e+00
## 5          30.58613  0.000000e+00
## 6          27.34675  0.000000e+00
## 7          13.58206  5.293554e-11
## 8          24.19846  0.000000e+00
```

The next step is to test the discrimination power of the combination of the elements chosen from the Greedy Wilks routine.

```
Group_mod2All<-dplyr::select(diamictdata.norm,Diamict_only,Si,Zr,b_star,Al,NG
R,MS,Ca,Rb)

# Select Training and Testing subsets -------------------------------------
data <- Group_mod2All
nColsF <- ncol(data)
set.seed(2969)
sample.ind = sample(2,
                    nrow(data),
                    replace = T,
                    prob = c(0.25,0.75))
data.test = data[sample.ind==1,]#data.dev
data.train = data[sample.ind==2,]#data.val

# #See how balanced the test & training sets look as Group; training set shou
ld be balanced
table(data$Diamict_only)/nrow(data)

##
##  clastrich_diamict    massive_diamict    mud_wdropstones
##        0.008433735        0.728915663        0.017349398
##      sandy_diamict stratified_diamict
##        0.090361446        0.154939759

table(data.test$Diamict_only)/nrow(data.test)

##
##  clastrich_diamict    massive_diamict    mud_wdropstones
##        0.006829268        0.728780488        0.021463415
##      sandy_diamict stratified_diamict
##        0.086829268        0.156097561
```

```
table(data.train$Diamict_only)/nrow(data.train)

##
##  clastrich_diamict    massive_diamict    mud_wdropstones
##            0.00896            0.72896            0.01600
##      sandy_diamict stratified_diamict
##            0.09152            0.15456

#if one of the training groups is too large, you need to resample using Caret
set.seed(9560)
dtrainCols <- ncol(data.train)

down_train <- downSample(x = data.train[, 1:dtrainCols],
                         y = data.train$Diamict_only)
table(down_train$Diamict_only)/nrow(down_train)

##
##  clastrich_diamict    massive_diamict    mud_wdropstones
##                0.2                0.2                0.2
##      sandy_diamict stratified_diamict
##                0.2                0.2

dataD.train <- down_train[,1:dtrainCols]

# Discriminant Testing ---------------------------------------------------
qda.fit2 <- qda(Diamict_only ~., data=dataD.train)
qda.fit2

## Call:
## qda(Diamict_only ~ ., data = dataD.train)
##
## Prior probabilities of groups:
##  clastrich_diamict    massive_diamict    mud_wdropstones
##                0.2                0.2                0.2
##      sandy_diamict stratified_diamict
##                0.2                0.2
##
## Group means:
##                           Si        Zr    b_star        Al       NGR
## clastrich_diamict  0.7368108 0.4128705 0.2628571 0.6715367 0.5119048
## massive_diamict    0.6658299 0.3635199 0.4085714 0.6735068 0.5297619
## mud_wdropstones    0.5509637 0.3769569 0.5961905 0.4964558 0.5238095
## sandy_diamict      0.6857384 0.4515262 0.3000000 0.7116534 0.5178571
## stratified_diamict 0.4298626 0.4997714 0.4576190 0.4888311 0.3630952
##                           MS        Ca        Rb
## clastrich_diamict  0.4041324 0.4482569 0.4118703
## massive_diamict    0.4296267 0.4479437 0.4420490
## mud_wdropstones    0.4429016 0.5430975 0.5388665
## sandy_diamict      0.3683626 0.3740992 0.4090718
## stratified_diamict 0.4203510 0.4663823 0.5702371
```

```
qda.class <- predict(qda.fit2, data.test)$class
qda.table <- table(qda.class, data.test$Diamict_only)
qda.table

##
## qda.class            clastrich_diamict massive_diamict mud_wdropstones
##   clastrich_diamict                  7              70               2
##   massive_diamict                    0             265               1
##   mud_wdropstones                    0              85              18
##   sandy_diamict                      0             207               0
##   stratified_diamict                 0             120               1
##
## qda.class            sandy_diamict stratified_diamict
##   clastrich_diamict              8                  4
##   massive_diamict               21                 32
##   mud_wdropstones                2                  5
##   sandy_diamict                 55                 37
##   stratified_diamict             3                 82

mean(qda.class == data.test$Diamict_only)

## [1] 0.4165854

fitControl <- trainControl(## 10-fold CV
  method = "repeatedcv",
  number = 10,
  ## repeated ten times
  repeats = 100)
set.seed(825)
qdaFit3 <- train(Diamict_only~ ., data = dataD.train,
                 method = "qda",
                 trControl = fitControl,
                 finalModel=TRUE,
                 verbose = FALSE,
                 na.action = na.omit)
qdaFit3

## Quadratic Discriminant Analysis
##
## 140 samples
##   8 predictor
##   5 classes: 'clastrich_diamict', 'massive_diamict', 'mud_wdropstones', 's
andy_diamict', 'stratified_diamict'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 100 times)
## Summary of sample sizes: 125, 125, 125, 126, 126, 129, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.5529401  0.4404742
```

```
qdaFit3$finalModel

## Call:
## qda(x, grouping = y, finalModel = TRUE, verbose = FALSE)
##
## Prior probabilities of groups:
##  clastrich_diamict    massive_diamict    mud_wdropstones
##                0.2                0.2                0.2
##      sandy_diamict stratified_diamict
##                0.2                0.2
##
## Group means:
##                           Si        Zr    b_star        Al       NGR
## clastrich_diamict  0.7368108 0.4128705 0.2628571 0.6715367 0.5119048
## massive_diamict    0.6658299 0.3635199 0.4085714 0.6735068 0.5297619
## mud_wdropstones    0.5509637 0.3769569 0.5961905 0.4964558 0.5238095
## sandy_diamict      0.6857384 0.4515262 0.3000000 0.7116534 0.5178571
## stratified_diamict 0.4298626 0.4997714 0.4576190 0.4888311 0.3630952
##                           MS        Ca        Rb
## clastrich_diamict  0.4041324 0.4482569 0.4118703
## massive_diamict    0.4296267 0.4479437 0.4420490
## mud_wdropstones    0.4429016 0.5430975 0.5388665
## sandy_diamict      0.3683626 0.3740992 0.4090718
## stratified_diamict 0.4203510 0.4663823 0.5702371

confusionMatrix(data.test$Diamict_only, predict(qdaFit3, data.test))

## Confusion Matrix and Statistics
##
##                     Reference
## Prediction           clastrich_diamict massive_diamict mud_wdropstones
##   clastrich_diamict                  7               0               0
##   massive_diamict                   70             265              85
##   mud_wdropstones                    2               1              18
##   sandy_diamict                      8              21               2
##   stratified_diamict                 4              32               5
##                     Reference
## Prediction           sandy_diamict stratified_diamict
##   clastrich_diamict              0                  0
##   massive_diamict              207                120
##   mud_wdropstones                0                  1
##   sandy_diamict                 55                  3
##   stratified_diamict            37                 82
##
## Overall Statistics
##
##                Accuracy : 0.4166
##                  95% CI : (0.3862, 0.4475)
##     No Information Rate : 0.3112
##     P-Value [Acc > NIR] : 7.497e-13
```

```
##
##                   Kappa : 0.1824
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: clastrich_diamict Class: massive_diamict
## Sensitivity                          0.076923                 0.8307
## Specificity                          1.000000                 0.3173
## Pos Pred Value                       1.000000                 0.3548
## Neg Pred Value                       0.917485                 0.8058
## Prevalence                           0.088780                 0.3112
## Detection Rate                       0.006829                 0.2585
## Detection Prevalence                 0.006829                 0.7288
## Balanced Accuracy                    0.538462                 0.5740
##                      Class: mud_wdropstones Class: sandy_diamict
## Sensitivity                         0.16364              0.18395
## Specificity                         0.99563              0.95317
## Pos Pred Value                      0.81818              0.61798
## Neg Pred Value                      0.90828              0.73932
## Prevalence                          0.10732              0.29171
## Detection Rate                      0.01756              0.05366
## Detection Prevalence                0.02146              0.08683
## Balanced Accuracy                   0.57963              0.56856
##                      Class: stratified_diamict
## Sensitivity                             0.3981
## Specificity                             0.9048
## Pos Pred Value                          0.5125
## Neg Pred Value                          0.8566
## Prevalence                              0.2010
## Detection Rate                          0.0800
## Detection Prevalence                    0.1561
## Balanced Accuracy                       0.6514
```

```
#relative importance of variables in model
qdaImp <- varImp(qdaFit3, scale = FALSE)
qdaImp
```

```
## ROC curve variable importance
##
##   variables are sorted by maximum importance across the classes
##        clastrich_diamict massive_diamict mud_wdropstones sandy_diamict
## b_star            0.9847          0.8246          0.9145        0.8246
## Si                0.8776          0.6709          0.9145        0.7270
## Al                0.8954          0.6365          0.7334        0.8202
## Rb                0.7360          0.5357          0.7934        0.6824
## NGR               0.5408          0.5325          0.7698        0.5325
## Zr                0.6148          0.6148          0.7041        0.6148
## Ca                0.6875          0.7015          0.5574        0.6735
## MS                0.5179          0.6065          0.5427        0.5153
##        stratified_diamict
## b_star              0.9847
## Si                  0.8776
## Al                  0.8954
## Rb                  0.7360
## NGR                 0.5408
## Zr                  0.5702
## Ca                  0.7015
## MS                  0.6065
```

Support Vector Machine Classification Analysis of Diamict-only Lithofacies

```
## svm >
svm.model <- svm(Diamict_only ~ ., data = dataD.train, cost = 100, gamma = 1)
svm.pred <- predict(svm.model, data.test)
## compute svm confusion matrix >
svmtable <- table(pred = svm.pred, data.test$Diamict_only)
svmtable
```

```
##
## pred                 clastrich_diamict massive_diamict mud_wdropstones
##   clastrich_diamict                  6              66               3
##   massive_diamict                    1             431               3
##   mud_wdropstones                    0              40              16
##   sandy_diamict                      0             126               0
##   stratified_diamict                 0              84               0
##
## pred                 sandy_diamict stratified_diamict
##   clastrich_diamict              6                  4
##   massive_diamict               31                 39
##   mud_wdropstones                0                  0
##   sandy_diamict                 48                 18
##   stratified_diamict             4                 99
```

```
classAgreement(svmtable,qda.table)
```

```
## $diag
## [1] 0.5853659
##
## $kappa
## [1] 0.3025034
##
## $rand
## [1] 0.5527782
##
## $crand
## [1] 0.144368
```

```
## compute rpart confusion matrix >
fitControl <- trainControl(## 10-fold CV
  method = "repeatedcv",
  number = 10,#10
  ## repeated ten times
  repeats = 100)#100
set.seed(645)
#fitControl <- trainControl(number = 200)
SVMFit2 <- train(Diamict_only~ ., data = dataD.train,
                 method = "svmRadial",
                 trControl = fitControl,
                 tuneLength = 12,
                 finalModel=TRUE,
                 verbose = FALSE,
                 scaled = FALSE,
                 na.action = na.omit)
SVMFit2

## Support Vector Machines with Radial Basis Function Kernel
##
## 140 samples
##   8 predictor
##   5 classes: 'clastrich_diamict', 'massive_diamict', 'mud_wdropstones', 's
andy_diamict', 'stratified_diamict'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 100 times)
## Summary of sample sizes: 125, 125, 125, 126, 127, 125, ...
## Resampling results across tuning parameters:
##
##   C       Accuracy   Kappa
##     0.25  0.2720610  0.1320918
##     0.50  0.3628651  0.2250427
##     1.00  0.4327305  0.3027396
##     2.00  0.4541064  0.3240749
##     4.00  0.4797717  0.3511361
##     8.00  0.5065166  0.3835791
##    16.00  0.5515361  0.4389786
##    32.00  0.5739535  0.4668211
##    64.00  0.5809287  0.4753673
##   128.00  0.5807713  0.4751457
##   256.00  0.5742982  0.4670105
##   512.00  0.5509512  0.4379128
##
## Tuning parameter 'sigma' was held constant at a value of 0.1616777
## Accuracy was used to select the optimal model using  the largest value.
## The final values used for the model were sigma = 0.1616777 and C = 64.

SVMFit2$finalModel

## Support Vector Machine object of class "ksvm"
##
```

```
## SV type: C-svc  (classification)
##  parameter : cost C = 64
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.161677737572634
##
## Number of Support Vectors : 115
##
## Objective Function Value : -1373.785 -649.626 -2035.449 -543.3263 -1449.35
5 -2145.998 -1597.825 -704.8161 -849.7802 -1277.091
## Training error : 0.3
```

```
confusionMatrix(data.test$Diamict_only, predict(SVMFit2, data.test))
```

```
## Confusion Matrix and Statistics
##
##                        Reference
## Prediction              clastrich_diamict massive_diamict mud_wdropstones
##   clastrich_diamict                     7               0               0
##   massive_diamict                     142             240              99
##   mud_wdropstones                       2               0              19
##   sandy_diamict                        17              10               0
##   stratified_diamict                    8              18               6
##                        Reference
## Prediction              sandy_diamict stratified_diamict
##   clastrich_diamict                 0                  0
##   massive_diamict                 169                 97
##   mud_wdropstones                   1                  0
##   sandy_diamict                    52                 10
##   stratified_diamict               35                 93
##
## Overall Statistics
##
##                Accuracy : 0.401
##                  95% CI : (0.3708, 0.4317)
##     No Information Rate : 0.2615
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.205
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: clastrich_diamict Class: massive_diamict
## Sensitivity                          0.039773                 0.8955
## Specificity                          1.000000                 0.3303
## Pos Pred Value                       1.000000                 0.3213
## Neg Pred Value                       0.833988                 0.8993
## Prevalence                           0.171707                 0.2615
## Detection Rate                       0.006829                 0.2341
## Detection Prevalence                 0.006829                 0.7288
## Balanced Accuracy                    0.519886                 0.6129
##                        Class: mud_wdropstones Class: sandy_diamict
## Sensitivity                           0.15323              0.20233
## Specificity                           0.99667              0.95182
## Pos Pred Value                        0.86364              0.58427
## Neg Pred Value                        0.89531              0.78098
## Prevalence                            0.12098              0.25073
## Detection Rate                        0.01854              0.05073
## Detection Prevalence                  0.02146              0.08683
## Balanced Accuracy                     0.57495              0.57708
##                        Class: stratified_diamict
## Sensitivity                               0.46500
## Specificity                               0.91879
## Pos Pred Value                            0.58125
## Neg Pred Value                            0.87630
## Prevalence                                0.19512
## Detection Rate                            0.09073
## Detection Prevalence                      0.15610
## Balanced Accuracy                         0.69189

#relative importance of variables in model
SVMImp <- varImp(SVMFit2, scale = FALSE)
SVMImp

## ROC curve variable importance
##
##   variables are sorted by maximum importance across the classes
##        clastrich_diamict massive_diamict mud_wdropstones sandy_diamict
## b_star            0.9847          0.8246          0.9145        0.8246
## Si                0.8776          0.6709          0.9145        0.7270
## Al                0.8954          0.6365          0.7334        0.8202
## Rb                0.7360          0.5357          0.7934        0.6824
## NGR               0.5408          0.5325          0.7698        0.5325
## Zr                0.6148          0.6148          0.7041        0.6148
## Ca                0.6875          0.7015          0.5574        0.6735
## MS                0.5179          0.6065          0.5427        0.5153
##        stratified_diamict
## b_star             0.9847
## Si                 0.8776
## Al                 0.8954
## Rb                 0.7360
## NGR                0.5408
## Zr                 0.5702
## Ca                 0.7015
## MS                 0.6065
```

02-Penkrot et al., 2018 Geosphere Unsupervised Classification: optimal-model results

Michelle Penkrot

3/19/2018

Code Information

This code will import physical property data and normalized scanning XRF elemental concentrations (NMS normalized; Lyle et al., 2012) from Integrated Ocean Drilling Site U1419, and then perform both mixture-model clustering and heirarchical clustering on these data. See here for more details about this drilling location: http://iodp.tamu.edu/scienceops/expeditions/alaska_tectonics_climate.html.

This file will only produce 3 principle components and five clusters, which is the optimal model output. See 03_Penkrot-2018-Geosphere-Unsupervised-Classification_general-model.Rmd for the general model that uses 2-3 PCs and up to five clusters. Note that the cluster number (e.g., Cluster 1, Cluster 2, etc.) produced by he code is arbitrary, so we renamed the clusters post-modeling in rank order so that Cluster 1 indicates the most common cluster and Cluster 5 the least common.

The mclust portion of the following code is modified from R scripts that were originally written and kindly supplied by Karl Ellefsen (2015, personal communication). See Ellefsen, K. J., Smith, D. B., & Horton, J. D. (2014). A modified procedure for mixture-model clustering of regional geochemical data. Applied Geochemistry, 51, 315–326. https://doi.org/10.1016/j.apgeochem.2014.10.011

Load Packages

```
library(knitr)
library(ezknitr)
library(rgr)
library(ggplot2)
library(psych)
library(parallel)
library(rrcov)
library(mclust)
library(lsr)
library(parallel)
```

Create functions

```
CalcSampleClusters <- function( theData, nPDFs, sampleSize, sampleSpace, nIte
r )
{
  require( mclust, quiet=TRUE )

  theLogLikelihoods <- vector( mode="numeric" )
  nErrors <- 0
  for( i in 1:nIter ) {
    S <- sample( sampleSpace, size = sampleSize )

    clusterResult <- tryCatch( Mclust( theData, nPDFs, modelNames=c("VVV"), i
nitialization=list(subset=S) ), error=function(e){ e } )

    if( inherits( clusterResult, "error" ) ) {
      nErrors <- nErrors + 1
    } else {
      theLogLikelihoods <- append( theLogLikelihoods, clusterResult$loglik )
      if( max( theLogLikelihoods ) == clusterResult$loglik ) {
        bestClusterResult <- clusterResult
      }
      if( min( theLogLikelihoods ) == clusterResult$loglik ) {
        worstClusterResult <- clusterResult
      }
    }
  }

  return( list( theLogLikelihoods=theLogLikelihoods, bestClusterResult=bestCl
usterResult,
                worstClusterResult=worstClusterResult, nErrors=nErrors ) )
}
```

Load data

Loads the data from a csv file and separates based on type (i.e. physical property or scanning XRF elemental). The physical property data are centered and scaled (z-score), and an isometric log ratio transformation (ILR) is performed on the elemental data to open them.

NOTE: This code will only analyze the diamict portion of the core. See 03_Penkrot-2018-Geosphere-Unsupervised-Classification_general-model.Rmd for code that analyzes entire data set.

```
data<-read.csv("../raw_data/2018-03-20_U1419-Penkrot_Geosphere-2018-data.csv"
)
data<-data[509:6490,] #cuts the data down to the diamict-only portion of the
core
lith<-data$Diamict_only_code
depth<-data$CCFS_A

physprops<-data[c("b_star","NGR","MS")]
elements<-data[c("Al","Ca","Rb","Zr","K","Si")]

physprops_scaled<-scale(physprops, center=TRUE,scale=TRUE) # Centers and scal
es the physical property data
elements_ilr<-ilr(as.matrix(elements)) # Performs isometric log ratio transfo
rmation to open XRF data
```

```
##   ** Are the data all in the same measurement units? **
```

Data Selection

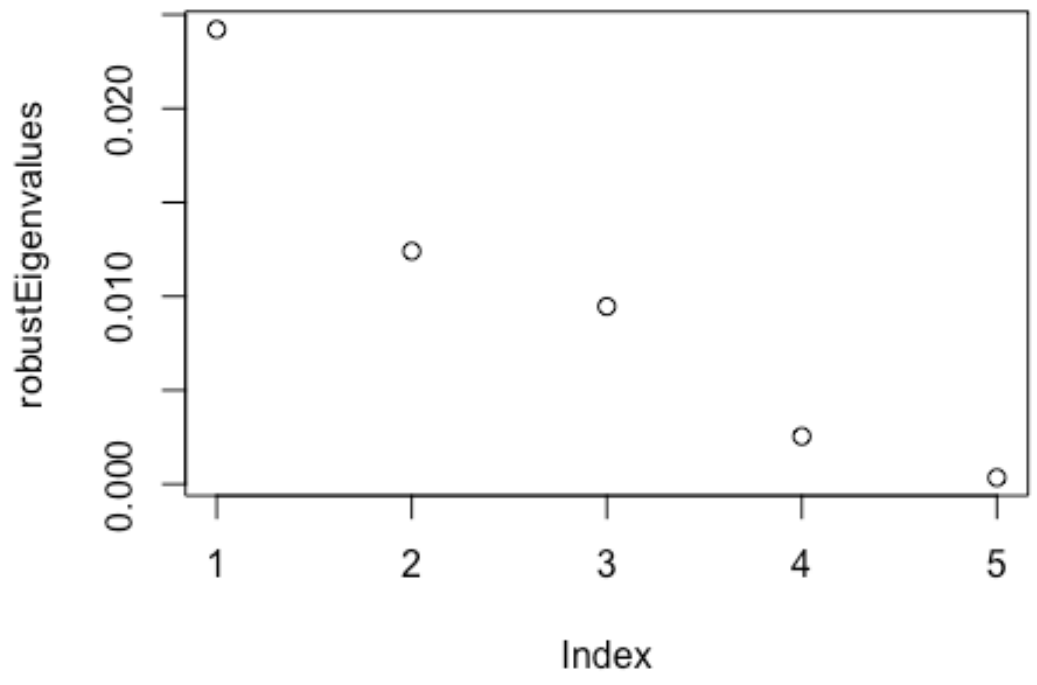
Select which data to include in cluster analysis. This study ran the model 3 times, with physical property and scanning XRF elemental data inputs (run name: G1), only physical property inputs (run name: G2) and only scanning XRF data (run name G3). The optimal data set was G3 and is used in the paper.

```
run.name<-"G3"
data_transformed<-as.matrix(cbind(elements_ilr)) #scanning XRF only; G3
```

Data Processing

Performs a robust principle component analysis on the input data. These principle component results are used in the cluster analysis rather than the raw data. The alpha value lets you select which percentage of the data to be excluded.

```
# select alpha(s): 0.98, 0.96, and 0.92, which will exclude 2%, 4%, and 8% of
the data, respectively
alpha <- c( 0.98)
outputDirectories <- c( "../produced_data/") # Change to appropriate output d
irectory

  mcdResult <- covMcd( data_transformed, alpha=alpha )
  robustIlrCenter <- mcdResult$center
  robustIlrCov <- mcdResult$cov
  centeredIlrCoefs <- data_transformed -  rep.int( 1, nrow( data_transformed
) ) %o% robustIlrCenter
  svdResult <- svd( robustIlrCov )
  robustEigenvectors <- svdResult$u
  robustEigenvalues <- svdResult$d   # namely, robust variances
  robustPCs <- centeredIlrCoefs %*% robustEigenvectors
  plot(robustEigenvalues) # Scree plot
```



```
  save(robustIlrCenter, robustEigenvectors, robustEigenvalues, robustPCs,
       file=paste( outputDirectories,run.name,"_PrinComp_3PC.dat", sep="" ) )
```

Model-based clustering (MClust)

Performs model clustering on robust principle component results. For the optimal model, we use three principles components (nPCs) and five clusters (nPDFs). This study ran the model cluster analysis for both 2 & 3 PCs and 2-5 clusters (see 03_Penkrot-2018-Geosphere-Unsupervised-Classification_general-model.Rmd)

```
nWorkers <- 4 # should be <= 4 on my machine, number of processor cores
cl <- makeCluster( nWorkers )
clusterSetRNGStream( cl, 123 )

outputDirectories <- c("../produced_data/") #Change to appropriate output dir
ectory

                       for( j in 1:length(outputDirectories) ) {


                       load( paste( outputDirectories[j],run.name,"_PrinComp_
3PC.dat", sep="" ) )
```

```
                       nRows <- nrow( robustPCs )
                       sampleSize <- as.integer( 0.75 * nRows )
                       sampleSpace <- 1:nRows

                       for( nPCs in 3:3) { #uses 3 principle components

                       for( nPDFs in 5:5 ) { #creates 5 clusters

                       clusterOutputDirectory <- paste( outputDirectories[j],
run.name,"_modelclust_",nPCs, "-PCs__", nPDFs, "-PDFs/", sep="" )
                       dir.create(clusterOutputDirectory)

                       nIter <- 100 + ( nPDFs - 2 ) * 150 + ( nPCs - 4 ) * 30
                       nIterPerWorker <- as.integer( nIter / nWorkers )
                       cat( sprintf( "No. of principal components:  %3d    No
. of pdfs: %3d    No. of iter: %3d    No. of iter per worker: %3d\n",
                       nPCs, nPDFs, nIter, nIterPerWorker ) )

                       tmpResult <-clusterCall( cl, CalcSampleClusters, robus
tPCs[,1:nPCs], nPDFs, sampleSize, sampleSpace, nIterPerWorker  )

                       theLogLikelihoods <- tmpResult[[1]]$theLogLikelihoods
                       bestClusterResult <- tmpResult[[1]]$bestClusterResult
                       worstClusterResult <- tmpResult[[1]]$worstClusterResul
t
                       nErrors <- tmpResult[[1]]$nErrors
                       for( i in 2:nWorkers ) {

                       theLogLikelihoods <- append( theLogLikelihoods, tmpRes
ult[[i]]$theLogLikelihoods )

                       if( bestClusterResult$loglik < tmpResult[[i]]$bestClus
terResult$loglik )
                       bestClusterResult <- tmpResult[[i]]$bestClusterResult

                       if( worstClusterResult$loglik > tmpResult[[i]]$worstCl
usterResult$loglik )
                       worstClusterResult <- tmpResult[[i]]$worstClusterResul
t

                       nErrors <- nErrors + tmpResult[[i]]$nErrors
                       }

                       save( theLogLikelihoods, bestClusterResult, worstClust
erResult, nErrors,
                       file=paste( clusterOutputDirectory,"data.dat", sep=""
) )

                       cat( sprintf( "No. of errors: %3d\n", nErrors ) )
                       }
                       }
                       }

## Warning in dir.create(clusterOutputDirectory): '../produced_data/
## G3_modelclust_3-PCs__5-PDFs' already exists

## No. of principal components:    3    No. of pdfs:   5    No. of iter: 520
No. of iter per worker: 130
## No. of errors:   0

                       stopCluster( cl )
```

Combines model clustering results into one matrix.

```
load(paste("../produced_data/G3_modelclust_3-PCs__5-PDFs/data.dat",sep=""))
bestclusterresult<-bestClusterResult$classification
mclust_results<-bestclusterresult
```

Hierarchical-based clustering (hclust)

Performs Hierarchical cluster analysis using the hclust function from the R stats package. Hierarchical clustering produces a dendrogram based on similariities of the downcore data. The dendrogram is cut off at heights that produces 5 clusters.

```
distance3<-dist(as.data.frame(robustPCs[,1:3]),method="euclidean") # uses 3 p
rinciple components
MyTree3<-hclust(distance3, method='ward.D2')
Cut3PC_5CL<-as.data.frame(cutree(MyTree3, k=5)) # cuts hclust tree at 5 clust
ers
hclust_results<-(Cut3PC_5CL)
colnames(hclust_results)<-c("hclust_3PCs-5Clusters")
```

Statistical Analysis of Results

Statistical parameters used to validate cluster results through comparison with downcore changes in observed lithofacies. Chi-squared test, Cramer's V-value, F-measure and Rand Index are calculated. This study only discusses the Chi-squared test and Cramer's V-value results.

```
results<-cbind(depth,mclust_results,hclust_results)
r<-length(results)-1
c<-4
statistics<-matrix(0,r,c) # r=number of models, c=number of statistical tests
for (i in 1:r){
  q<-results[,i+1]
  tbl<-table(lith,q)
  chi<-chisq.test(tbl)
  cv<-cramersV(tbl)
  x<-chi$statistic
  y<-chi$parameter
  z<-chi$p.value
  statistics[i,]<-cbind(x,y,z,cv)
}

## Warning in chisq.test(tbl): Chi-squared approximation may be incorrect

## Warning in chisq.test(...): Chi-squared approximation may be incorrect

## Warning in chisq.test(tbl): Chi-squared approximation may be incorrect

## Warning in chisq.test(...): Chi-squared approximation may be incorrect

colnames(statistics)<-c("X-square","Df","p-value","Cramers V Value")
nCols <- ncol(results)
rownames(statistics)<-colnames(results[,2:nCols])
```

Save Model Output

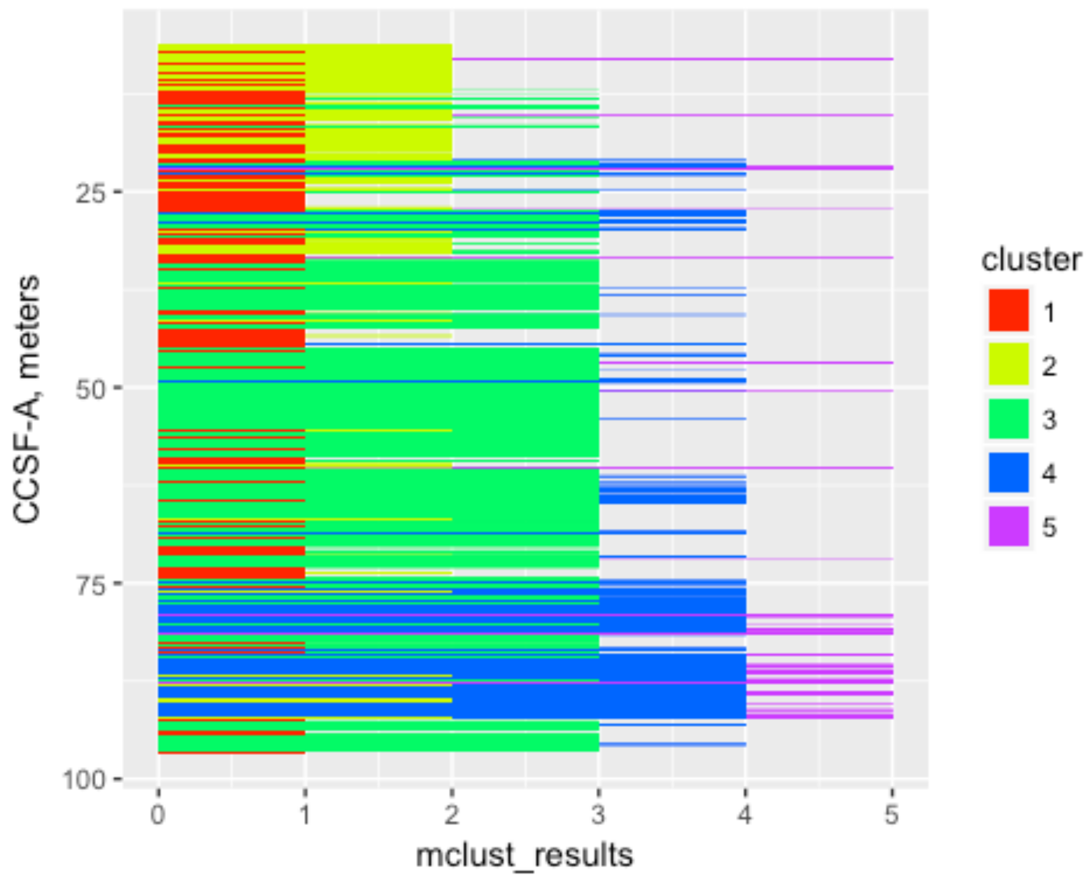
Saves clustering results from both model and hierarchical clustering for 2-3 PCs and 2-5 clusters, and statistical parameter results as .csv files.

```
write.csv(results,file=paste("../produced_data/",toString(run.name),"_3PC-5cl
usteringresults.csv",sep="")) # Saves Hierarchical and model clustering resul
ts
write.csv(statistics,file=paste("../produced_data/",toString(run.name),"_3PC-
5clsuter_statistics.csv",sep="")) # Saves statistical validation results
```

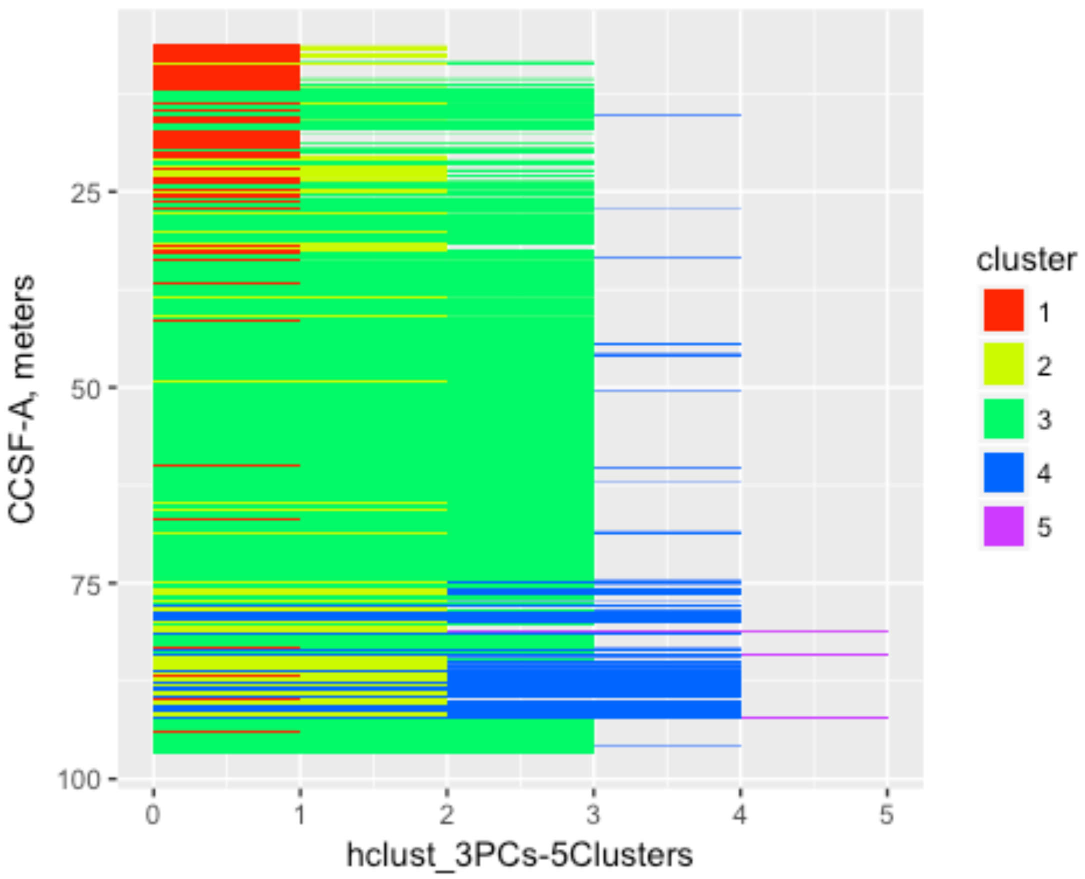
Create plots of cluster models

Plots of downcore distributions of cluster results for both model and hierarchical clustering, 2-3 PCs and 2-5 clusters. Plots go in "plot" RProject folder.

```
titles<-colnames(results)
resultsd<-results[!duplicated(results$depth),]
nColsL <- ncol(resultsd)
for(i in 2:nColsL){
  cluster<-factor(resultsd[,i])
  p<-ggplot(resultsd, aes(x=resultsd$depth,y=resultsd[,i],fill=cluster))+geom
_bar(stat="identity",size=3,width=0.1) +
    scale_fill_manual(values=rainbow(n=length(unique(resultsd[,i])))) + coord
_flip() + scale_x_reverse() +
    labs(x="CCSF-A, meters",y=toString(colnames(resultsd)[i]))
  print(p)
  ggsave(filename=paste("../plot/",toString(run.name),"_",toString(colnames(r
esultsd)[i]),"_optimal-results.pdf",sep=''),height=12,width=3)
}

## Warning: position_stack requires non-overlapping x intervals


## Warning: position_stack requires non-overlapping x intervals
```



```
## Warning: position_stack requires non-overlapping x intervals

## Warning: position_stack requires non-overlapping x intervals
```



CCSF-A, meters
25
50
75
100
0
1
2
3
4
5
hclust_3PCs-5Clusters
cluster
1
2
3
4
5

03-Penkrot et al., 2018 Geosphere U1419 general cluster analysis

Michelle Penkrot

3/19/2018

Code Information

This code will import physical property data and normalized scanning XRF elemental concentrations (NMS normalized; Lyle et al., 2012) from Integrated Ocean Drilling Site U1419, and then perform both mixture-model clustering and heirarchical clustering on these data. See here for more details about this drilling location: http://iodp.tamu.edu/scienceops/expeditions/alaska_tectonics_climate.html.

This file will produce 2-3 principle components and one to five clusters, which is the general model output. The user should be aware that running the full model (this code) takes considerable computing time. It is optimized to use parallel processing on four cores. Running the mclust routine for five clusters with two and three principle components can take up to 24 hours.

Note that the cluster number (e.g., Cluster 1, Cluster 2, etc.) produced by he code is arbitrary, so we renamed the clusters post-modeling in rank order so that Cluster 1 indicates the most common cluster and Cluster 5 the least common.

The mclust portion of the following code is modified from R scripts that were originally written and kindly supplied by Karl Ellefsen (2015, personal communication). See Ellefsen, K. J., Smith, D. B., & Horton, J. D. (2014). A modified procedure for mixture-model clustering of regional geochemical data. Applied Geochemistry, 51, 315–326. https://doi.org/10.1016/j.apgeochem.2014.10.011

Load Packages

```
library(knitr)
library(rgr)
library(ggplot2)
library(psych)
library(parallel)
library(rrcov)
library(mclust)
library(lsr)
library(parallel)
```

Create functions

```
CalcSampleClusters <- function( theData, nPDFs, sampleSize, sampleSpace, nIte
r )
{
  require( mclust, quiet=TRUE )

  theLogLikelihoods <- vector( mode="numeric" )
  nErrors <- 0
```

```r
  for( i in 1:nIter ) {
    S <- sample( sampleSpace, size = sampleSize )

    clusterResult <- tryCatch( Mclust( theData, nPDFs, modelNames=c("VVV"), i
nitialization=list(subset=S) ), error=function(e){ e } )

    if( inherits( clusterResult, "error" ) ) {
      nErrors <- nErrors + 1
    } else {
      theLogLikelihoods <- append( theLogLikelihoods, clusterResult$loglik )
      if( max( theLogLikelihoods ) == clusterResult$loglik ) {
        bestClusterResult <- clusterResult
      }
      if( min( theLogLikelihoods ) == clusterResult$loglik ) {
        worstClusterResult <- clusterResult
      }
    }
  }

  return( list( theLogLikelihoods=theLogLikelihoods, bestClusterResult=bestCl
usterResult,
                worstClusterResult=worstClusterResult, nErrors=nErrors ) )
}
```

Load Data

Loads the data from a csv file. Separates data based on type (i.e. physical property or scanning XRF elemental). The physical property data are centered and scaled (z-score), and an isometric log ratio transformation (ILR) is performed on the elemental data to open them.

NOTE: See comments below to set the code to run on the full mud-diamict dataset. It is set here to run only on the diamict-only portion of the data.

```r
data<-read.csv("../raw_data/2018-03-20_U1419-Penkrot_Geosphere-2018-data.csv"
)
data<-data[509:6490,] #cuts the data down to the diamict-only portion of the
core; comment this line if you want to run the analyses on the full mud-diami
ct data set.
#lith<-data$Mud_Diamict_code
lith<-data$Diamict_only_code #comment this line and uncomment previous line i
fyou want to run the analyses on the full mud-diamict data set.
depth<-data$CCFS_A

physprops<-data[c("b_star","NGR","MS")]
elements<-data[c("Al","Ca","Rb","Zr","K","Si")]

physprops_scaled<-scale(physprops, center=TRUE,scale=TRUE) # Centers and scal
es the physical property data
```

```
elements_ilr<-ilr(as.matrix(elements)) # Performs isometric log ratio transfo
rmation to open XRF data

##   ** Are the data all in the same measurement units? **
```

Select Data for Models

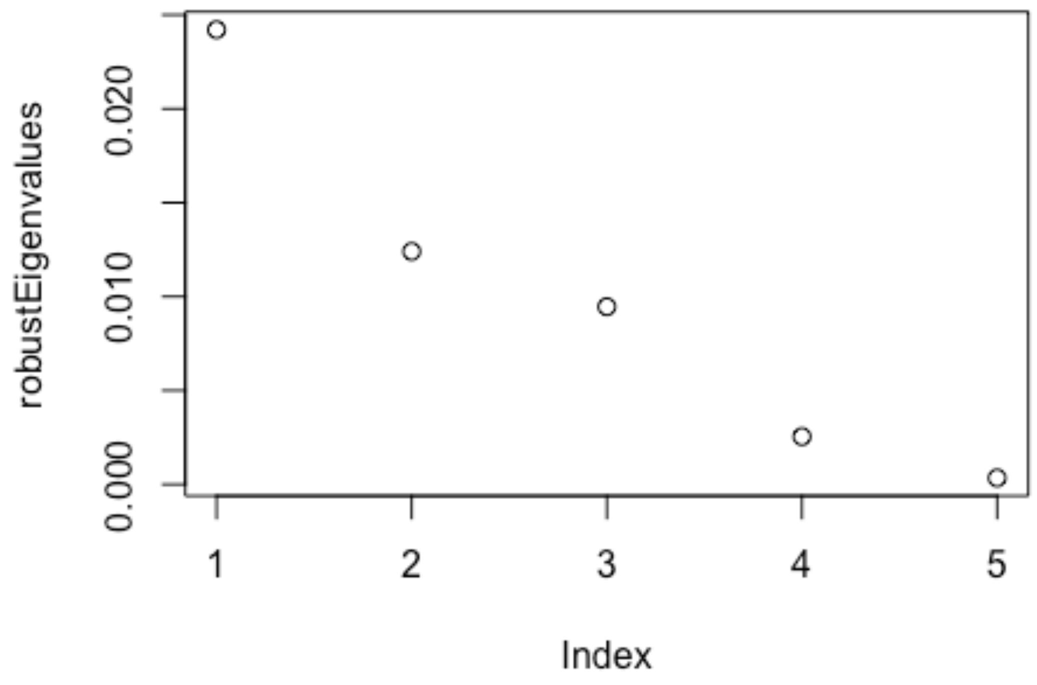
Select which data to include in cluster analysis. This study ran the model 3 times, with physical property and scanning XRF elemental data inputs (run name: G1), only physical property inputs (run name: G2) and only scanning XRF data (run name G3). The optimal data set was G3 and is used in the paper.

```
#run.name<-"G1" # Change for each cluster model run
#run.name<-"G2" # Change for each cluster model run
run.name<-"G3" # Change for each cluster model run

#data_transformed<-as.matrix(cbind(physprops_scaled,elements_ilr)) #phys prop
s & scanning XRF data; G1
#data_transformed<-as.matrix(cbind(physprops_scaled)) #phys props only; G2
data_transformed<-as.matrix(cbind(elements_ilr)) #scanning XRF only; G3
```

Data Processing

Performs a robust principle component analysis on the input data to remove outliers. These principle component results are used in the cluster analysis rather than the raw data. The alpha value lets you select which percentage of the data to be excluded as outliers. Follows this method: Martin Maechler, Peter Rousseeuw, Christophe Croux, Valentin Todorov, Andreas Ruckstuhl, Matias Salibian-Barrera, Tobias Verbeke, Manuel Koller, Eduardo L. T. Conceicao and Maria Anna di Palma (2016). robustbase: Basic Robust Statistics R package version 0.92-7. URL http://CRAN.R-project.org/package=robustbase

```
# select alpha(s): 0.98, 0.96, and 0.92, which will exclude 2%, 4%, and 8% of
the data as outliers, respectively
alpha <- c( 0.98)
outputDirectories <- c( "../produced_data/") # Change to appropriate output d
irectory

  mcdResult <- covMcd( data_transformed, alpha=alpha )
  robustIlrCenter <- mcdResult$center
  robustIlrCov <- mcdResult$cov
  centeredIlrCoefs <- data_transformed -  rep.int( 1, nrow( data_transformed
) ) %o% robustIlrCenter
  svdResult <- svd( robustIlrCov )
  robustEigenvectors <- svdResult$u
  robustEigenvalues <- svdResult$d   # namely, robust variances
  robustPCs <- centeredIlrCoefs %*% robustEigenvectors
  plot(robustEigenvalues) # Scree plot
```



```
  save(robustIlrCenter, robustEigenvectors, robustEigenvalues, robustPCs,
       file=paste( outputDirectories,run.name,"_PrinComp.dat", sep="" ) )
```

Model-based clustering (MClust)

Performs model clustering on robust principle component results. The number of principles components (nPCs) can be changed in line 121 and the number of clusters (nPDFs) can be changed in line 123. This study ran the model cluster analysis for both 2 & 3 PCs and 2-5 clusters.

```
nWorkers <- 4                     # should be <= 4 on my machine, i.e., number
of processor cores
cl <- makeCluster( nWorkers )
clusterSetRNGStream( cl, 123 )

outputDirectories <- c("../produced_data/") #Change to appropriate output dir
ectory

                        for( j in 1:length(outputDirectories) ) {
                        #for( j in 1:1 ) {

                        load( paste( outputDirectories[j],run.name,"_PrinComp.
dat", sep="" ) )
```

```
                        nRows <- nrow( robustPCs )
                        sampleSize <- as.integer( 0.75 * nRows )
                        sampleSpace <- 1:nRows

                        for( nPCs in 3:3) { #uses 2 & 3 principle components

                        for( nPDFs in 5:5 ) { #creates 2-5 clusters

                        clusterOutputDirectory <- paste( outputDirectories[j],
run.name,"_modelclust_",nPCs, "-PCs__", nPDFs, "-PDFs/", sep="" )
                        dir.create(clusterOutputDirectory)

                        nIter <- 100 + ( nPDFs - 2 ) * 150 + ( nPCs - 4 ) * 30
                        nIterPerWorker <- as.integer( nIter / nWorkers )
                        cat( sprintf( "No. of principal components:  %3d    No
. of pdfs: %3d    No. of iter: %3d    No. of iter per worker: %3d\n",
                        nPCs, nPDFs, nIter, nIterPerWorker ) )

                        tmpResult <-clusterCall( cl, CalcSampleClusters, robus
tPCs[,1:nPCs], nPDFs, sampleSize, sampleSpace, nIterPerWorker  )

                        theLogLikelihoods <- tmpResult[[1]]$theLogLikelihoods
                        bestClusterResult <- tmpResult[[1]]$bestClusterResult
                        worstClusterResult <- tmpResult[[1]]$worstClusterResul
t
                        nErrors <- tmpResult[[1]]$nErrors
                        for( i in 2:nWorkers ) {

                        theLogLikelihoods <- append( theLogLikelihoods, tmpRes
ult[[i]]$theLogLikelihoods )

                        if( bestClusterResult$loglik < tmpResult[[i]]$bestClus
terResult$loglik )
                        bestClusterResult <- tmpResult[[i]]$bestClusterResult

                        if( worstClusterResult$loglik > tmpResult[[i]]$worstCl
usterResult$loglik )
                        worstClusterResult <- tmpResult[[i]]$worstClusterResul
t

                        nErrors <- nErrors + tmpResult[[i]]$nErrors
                        }

                        save( theLogLikelihoods, bestClusterResult, worstClust
erResult, nErrors,
                        file=paste( clusterOutputDirectory,"data.dat", sep=""
) )
```

```
                        cat( sprintf( "No. of errors: %3d\n", nErrors ) )
                        }
                        }
                        }

                        stopCluster( cl )
```

Combines model clustering results into one matrix.

```
pc<-c(2,2,2,2,3,3,3,3)
pdf<-c(2,3,4,5,2,3,4,5)

mclust_results<-as.data.frame(matrix(0,nrow(data_transformed),length(pc)))
for(i in 1:length(pc)){
load(paste("../produced_data/",run.name,"_modelclust_",pc[1],"-PCs__",pdf[1],
"-PDFs/data.dat",sep=""))
bestclusterresult<-bestClusterResult$classification
mclust_results[,i]<-bestclusterresult
colnames(mclust_results)[i]<-paste("mclust_",pc[i],"PCs_",pdf[i],"Clusters",s
ep="")
}

## Warning in readChar(con, 5L, useBytes = TRUE): cannot open compressed file
## '../produced_data/G3_modelclust_2-PCs__2-PDFs/data.dat', probable reason
## 'No such file or directory'

## Error in readChar(con, 5L, useBytes = TRUE): cannot open the connection

load(paste("../produced_data/G3_modelclust_3-PCs__5-PDFs/data.dat",sep=""))
bestclusterresult<-bestClusterResult$classification
mclust_results<-bestclusterresult
```

Hierarchical-based clustering (hclust)

Performs Hierarchical cluster analysis using the hclust function from the R stats package. Hierarchical clustering produces a dendrogram based on similariities of the downcore data. The dendrogram is cut off at heights that produces 2-5 clusters.

```
distance2<-dist(as.data.frame(robustPCs[,1:2]),method="euclidean") # uses 2 p
rinciple components
MyTree2<-hclust(distance2, method='ward.D2')
Cut2PC_2CL<-as.data.frame(cutree(MyTree2,k=2)) # cuts hclust tree at 2 cluste
rs
Cut2PC_3CL<-as.data.frame(cutree(MyTree2, k=3))
Cut2PC_4CL<-as.data.frame(cutree(MyTree2, k=4))
Cut2PC_5CL<-as.data.frame(cutree(MyTree2, k=5)) # cuts hclust tree at 5 clust
ers

distance3<-dist(as.data.frame(robustPCs[,1:3]),method="euclidean") # uses 3 p
rinciple components
MyTree3<-hclust(distance3, method='ward.D2')
```

```
Cut3PC_2CL<-as.data.frame(cutree(MyTree3,k=2)) # cuts hclust tree at 2 cluste
rs
Cut3PC_3CL<-as.data.frame(cutree(MyTree3, k=3))
Cut3PC_4CL<-as.data.frame(cutree(MyTree3, k=4))
Cut3PC_5CL<-as.data.frame(cutree(MyTree3, k=5)) # cuts hclust tree at 5 clust
ers

hclust_results<-cbind(Cut2PC_2CL,Cut2PC_3CL,Cut2PC_4CL,Cut2PC_5CL,Cut3PC_2CL,
Cut3PC_3CL,Cut3PC_4CL,Cut3PC_5CL)
colnames(hclust_results)<-c("hclust_2PCs-2Clusters","hclust_2PCs-3Clusters","
hclust_2PCs-4Clusters","hclust_2PCs-5Clusters","hclust_3PCs-2Clusters","hclus
t_3PCs-3Clusters","hclust_3PCs-4Clusters","hclust_3PCs-5Clusters")

hclust_results<-(Cut3PC_5CL)
colnames(hclust_results)<-c("hclust_3PCs-5Clusters")
```

Statistical Analysis of Results

Statistical parameters used to validate cluster results through comparison with downcore changes in observed lithofacies. Chi-squared test, Cramer's V-value, F-measure and Rand Index are calculated. This study only discusses the Chi-squared test and Cramer's V-value results.

```
results<-cbind(depth,mclust_results,hclust_results)
r<-length(results)-1
c<-4
statistics<-matrix(0,r,c) # r=number of models, c=number of statistical tests
for (i in 1:r){
  q<-results[,i+1]
  tbl<-table(lith,q)
  chi<-chisq.test(tbl)
  cv<-cramersV(tbl)
  x<-chi$statistic
  y<-chi$parameter
  z<-chi$p.value
  statistics[i,]<-cbind(x,y,z,cv)
}

## Warning in chisq.test(tbl): Chi-squared approximation may be incorrect

## Warning in chisq.test(...): Chi-squared approximation may be incorrect

## Warning in chisq.test(tbl): Chi-squared approximation may be incorrect

## Warning in chisq.test(...): Chi-squared approximation may be incorrect

colnames(statistics)<-c("X-square","Df","p-value","Cramers V Value")
nCols <- ncol(results)
rownames(statistics)<-colnames(results[,2:nCols])
```

Save Model Output

Saves clustering results from both model and hierarchical clustering for 2-3 PCs and 2-5 clusters, and statistical parameter results as .csv files.

```
write.csv(results,file=paste("../produced_data/",toString(run.name),"_cluster
ingresults.csv",sep="")) # Saves Hierarchical and model clustering results
write.csv(statistics,file=paste("../produced_data/",toString(run.name),"_stat
istics.csv",sep="")) # Saves statistical validation results
```

Create plots of cluster models

Plots of downcore distributions of cluster results for both model and hierarchical clustering, 2-3 PCs and 2-5 clusters. Plots go in "plot" RProject folder.

```
titles<-colnames(results)
resultsd<-results[!duplicated(results$depth),]
nColsL <- ncol(resultsd)
for(i in 2:nColsL){
  cluster<-factor(resultsd[,i])
  p<-ggplot(resultsd, aes(x=resultsd$depth,y=resultsd[,i],fill=cluster))+geom
_bar(stat="identity",size=3,width=0.1) +
    scale_fill_manual(values=rainbow(n=length(unique(resultsd[,i])))) + coord
_flip() + scale_x_reverse() +
    labs(x="CCSF-A, meters",y=toString(colnames(resultsd)[i]))
  print(p)
  ggsave(filename=paste("../plot/",toString(run.name),"_",toString(colnames(r
esultsd)[i]),".pdf",sep=''),height=12,width=3)
  dev.off()
}

## Warning: position_stack requires non-overlapping x intervals

## Warning: position_stack requires non-overlapping x intervals

## Warning: position_stack requires non-overlapping x intervals

## Warning: position_stack requires non-overlapping x intervals
```